Bestimmung der semantischen Eigenschaften von Datenstromsystemen durch Black-Box-Tests

Frank Lauterwald, Niko Pollner, Klaus Meyer-Wegener
Lehrstuhl für Informatik 6 (Datenmanagement)
Friedrich-Alexander-Universität Erlangen-Nürnberg
{frank.lauterwald, niko.pollner, klaus.meyer-wegener}@fau.de

Abstract: Die Semantik von Datenstromsystemen (DSS) ist bislang nicht standardisiert. Für Anwendungsentwickler ist es jedoch wichtig zu wissen, wie sich ein bestimmtes System in einer bestimmten Situation verhält. Ebenso bedeutsam ist das Verhalten für föderierte Datenstromsysteme, die Anfragen automatisch auf verschiedene DSS verteilen. Als Hilfsmittel zur Beschreibung können semantische Modelle dienen. Diese werden parametrisiert und können durch verschiedene Parameterwerte das Verhalten verschiedener Systeme nachbilden. Da bisher auch kein allgemein anerkanntes Modell zur Beschreibung von DSS existiert, muss man sich möglicherweise mit verschiedenen Modellen auseinandersetzen. Daher wäre es hilfreich, die Bestimmung der jeweiligen Parameterwerte weitgehend zu automatisieren, wozu dieser Beitrag eine geeignete Evaluationsumgebung vorstellt. Diese vergleicht die Ausgaben eines DSS mit allen Vorhersagen, die ein Modell für verschiedene Parameter machen kann. Stimmen die Ergebnisse überein, sind die Parameter gefunden. Erfahrungen damit und Beschränkungen dieses Ansatzes werden diskutiert.

1 Einleitung und Motivation

Datenstromsysteme (DSS) finden zunehmenden Einsatz in realen Szenarien. Jedoch mangelt es bisher an ihrer Standardisierung, wodurch die Interoperabilität eingeschränkt und die Portierung von Anwendungen erschwert wird.

Die DSS unterscheiden sich insbesondere bezüglich der bereitgestellten Funktionalität, der Anfragesprachen und des Verhaltens.

Es ist zu erwarten, dass die fehlende Standardisierung aufgrund des zunehmenden Einsatzes von Datenstromsystemen in Zukunft ein immer größeres Problem darstellt. Allerdings zeigen sich in diesem Bereich auch einige Hoffnungsschimmer: Zum einen ist eine beginnende Konsolidierung im Markt der kommerziellen DSS zu sehen. Zum zweiten wurden in der Forschung bereits einige semantische Modelle entwickelt (z.B. [BDD+10]), die das Verhalten verschiedener DSS in einer gemeinsamen Terminologie beschreiben können. Zum dritten verspricht die Entwicklung föderierter DSS (z.B. [BCD+09]), die Unterschiede zwischen verschiedenen Systemen hinter einer gemeinsamen Schnittstelle zu verbergen.

Dieser Artikel vereint die letzten beiden Punkte, also semantische Modelle und föderierte DSS. Die behandelte Fragestellung ergibt sich aus den Erfahrungen der Autoren im von

ihnen durchgeführten DSAM-Projekt [DLF+10]. Der Data Stream Application Manager (DSAM) ist ein föderiertes DSS, das globale Anfragen auf ein Netzwerk heterogener DSS verteilt. Die Verteilungsentscheidung wird durch einen kostenbasierten Optimierer getroffen. Der Anwendungsentwickler wird somit von der Aufgabe befreit, selbst zu entscheiden, welche Anfrageteile auf welchem System ausgeführt werden, wodurch er aber auch keinen direkten Einfluss auf die Verteilung mehr hat. Dennoch muss sichergestellt werden, dass eine Anfrage die von ihm gewünschten Ergebnisse liefert. Eine formale Beschreibung des Verhaltens der verwendeten Systeme ist dazu unabdingbar. Der Beitrag dieses Artikels besteht aus einer Methode zur teilautomatischen Bestimmung des Verhaltens von DSS sowie einer Evaluation dieser Methodik am Beispiel des SECRET-Modells [BDD+10].

2 Methode

Black-Box-Tests haben sich bereits bei der Perfomance-Analyse von DSS bewährt [DLB+11]. Da für viele DSS kein Quellcode verfügbar ist, sind andere Methoden nicht anwendbar. Die Black-Box-Methode soll nun auch zur automatischen Bestimmung der semantischen Eigenschaften von DSS genutzt werden. Diese werden dabei durch Parameterwerte innerhalb eines semantischen Modells beschrieben. Als semantische Modelle werden hier solche Modelle bezeichnet, die das Verhalten von DSS zumindest in Teilaspekten beschreiben können und damit Vorhersagen über deren Anfrageergebnisse (für bekannte Eingabedaten und Anfragen) zulassen. Parametrisierbare Modelle können durch die Wahl geeigneter Parameterwerte die Anfrageergebnisse verschiedener DSS vorhersagen. Einzelne Aspekte des Systemverhaltens werden als Dimensionen bezeichnet, die idealerweise durch unabhängige Parameter beschrieben werden. Das hier beispielhaft verwendete SECRET-Modell ist ein solches Modell, das sich durch hohe Ausdrucksmächtigkeit auszeichnet. Es definiert Parameter in vier verschiedenen Dimensionen: *ScopE* beschreibt, über welche Zeitabschnitte Fensterinstanzen definiert sind; *Content* definiert, welche Tupel jeweils in diesen Fensterinstanzen enthalten sind; *Tick* gibt an, aufgrund welcher Ereignisse ein System aktiv wird (z.B. Eintreffen eines Tupels oder Fortschreiten der Zeit); *REport* schliesslich erlaubt die Angabe weiterer Kriterien, die erfüllt sein müssen, damit das System eine Ausgabe erzeugt.

Der Aufwand für die Parameterbestimmung steigt mit der Anzahl betrachteter DSS und Modelle. Daher sollte die Parameterbestimmung möglichst automatisch erfolgen.

Die dazu unternommenen Schritte werden im Folgenden beschrieben. Die erzielten Ergebnisse werden dann in Abschnitt 3 erläutert.

2.1 Simulator

Zunächst wurde ein Simulator in Python entwickelt. Dieser ist im Kern ein minimales DSS, das sich durch Konfiguration so verhalten kann wie jedes durch SECRET beschreibbare DSS. Ein- und Ausgabe erfolgt durch einfache CSV-Dateien. Da die nachfolgenden

Schritte von der Korrektheit des Simulators abhängen, muss er zunächst validiert werden. Dazu können die Testdaten aus [BDD+10] verwendet werden, sowie die Ausgaben des Simulators mit denen realer DSS verglichen werden.

2.2 Anfragen und Testdaten

Das SECRET-Modell befasst sich mit dem Verhalten zeitbasierter Fenster. Es kann erklären, wann Fenster ausgewertet werden und welche Tupel sie zu diesem Zeitpunkt enthalten. Daher wurde die einfachst mögliche Testanfrage verwendet, mittels derer diese Frage beantwortet werden kann:

```
SELECT min(id), max(id) FROM InputStream [keep 3 seconds]
```
[1]

Diese Anfrage gibt die niedrigste und höchste id der Tupel in einem Fenster aus. Verwendet man nun Testdaten mit aufsteigenden ids, so geht aus dem Anfrageergebnis eindeutig hervor, welche Tupel zum Auswertungszeitpunkt in einem Fenster waren.

Die Testdaten werden iterativ erzeugt: Der Simulator wird mit allen möglichen Parameterkombinationen konfiguriert und die Anfrageergebnisse werden aufgezeichnet. Führen zwei oder mehr Parameterkombinationen zu identischen Ergebnissen, so werden gezielt Testdaten hinzugefügt, die diese Kombinationen unterscheidbar machen sollen. Ist dies nicht möglich, wird der Grund dafür analysiert. Dieses Vorgehen wird wiederholt, bis alle unterscheidbaren Kombinationen auch unterscheidbare Ergebnisse liefern und für alle nicht-unterscheidbaren ein Grund angegeben werden kann. Ein Nachteil dieses Vorgehens ist, dass es möglicherweise eine große Anzahl an Testdatensätzen erfordert. Günstiger wäre es, die einzelnen Dimensionen des Modells auch einzeln testen zu können. Dies ist allerdings schwierig: Beispielsweise können bestimmte Parameterwerte in verschiedenen Dimensionen zur Unterdrückung von Ausgaben führen. Somit liefert ein Test in einer Dimension kein Ergebnis, wenn die Ergebnisse durch den Wert einer anderen Definition unterdrückt werden.

2.3 Parameterbestimmung von Datenstromsystemen

Um die semantischen Eigenschaften realer DSS zu testen, wird die Testanfrage auf diesen mit allen im vorherigen Schritt erzeugten Testdaten ausgeführt und die Ergebnisse werden aufgezeichnet. Diese Ergebnisse werden dann mit den Ausgaben des Simulators für alle möglichen Parameterkombinationen verglichen. Die Ausgabe des realen Systems sollte mit der Ausgabe des Simulators für genau eine Parameterkombination identisch sein. Die semantischen Parameter des realen Systems lassen sich unmittelbar daraus ablesen, mit *welchen* Parameterwerten der Simulator identische Ergebnisse erzeugt.

[1] Natürlich muss diese Anfrage in die jeweiligen Anfragesprachen der getesteten Systeme übersetzt werden.

3 Ergebnisse

Im diesem Abschnitt werden die Ergebnisse der Evaluation der vorgestellten Methoden auf Basis des SECRET-Modells beschrieben.

3.1 Modell und Simulator

Die Ergebnisse aus [BDD+10] konnten mit einigen Abweichungen manuell nachvollzogen werden.[2] Diese Ergebnisse dienen als Grundlage für die weiteren Versuche. Die automatische Parameterbestimmung soll also für jedes DSS (z.B. Coral8) die gleichen Ergebnisse liefern wie dieser manuelle Vorversuch.

Der Simulator wurde mittels der in [BDD+10] beschriebenen Testdaten validiert. Es wurde also getestet, ob der Simulator die in [BDD+10] beschriebenen Ergebnisse liefert. Als zweiten Validierungsschritt wurde der Simulator mit den bekannten Parametern für die in [BDD+10] beschriebenen realen DSS konfiguriert und es wurde getestet, ob er dieselben Ergebnisse erzeugt wie diese Systeme. Mit diesem Test wurde sichergestellt, dass der Simulator die Ergebnisse realer DSS korrekt simuliert.

3.2 Testdaten

Ein Ziel war es, Testdaten zu erzeugen, die für jede Ausprägung von SECRET-Parametern unterschiedliche Anfrageergebnisse liefern. SECRET unterscheidet drei Werte für *Tick* und 16 Kombinationen der *REport*-Flags. Für *ScopE* sind beliebige Formeln möglich [3], allerdings verwenden die in [BDD+10] beschriebenen Systemen nur 2 unterschiedliche Formeln. Daher wurden auch nur diese in die Untersuchung einbezogen. Durch Ausmultiplizieren ergibt sich eine Maximalzahl von 96 unterschiedlichen Parameterkombinationen.

Neben den SECRET-Parametern sind weitere Details der Anfrage zu spezifizieren (z.B. die Fenstergröße). Diese werden zu *Konfigurationen* zusammengefasst. Die Versuche wurden in zwei Konfigurationen, T_1 und T_2 durchgeführt, die sich im Wert des Slide-Parameters und der Häufigkeit von Ausgaben unterscheiden. [4] Die Werte in T_1 wurden für größtmögliche Unterscheidungskraft gewählt. So sind 84 der 96 möglichen Klassen anhand ihrer Ausgaben voneinander unterscheidbar. Allerdings sind bei vielen Systemen weder Slide noch Reporting Frequency frei wählbar. In T_2 sind diese Werte daher derart belegt, dass alle betrachteten Systeme mit ihr konfiguriert werden können. Hierdurch sinkt die Ausdrucksmächtigkeit deutlich und es können nur noch 15 Klassen unterschieden wer-

[2] Insbesondere verhielt sich in unseren Versuchen StreamBase zeitgetrieben und Coral8 konnte nicht batch-, sondern nur tupelgetrieben betrieben werden. Außerdem schien Coral8 nur für $\lambda = 1$ auf *non-empty content* zu reagieren. Solche Abweichungen stellen für die vorliegende Arbeit kein Problem dar.

[3] *ScopE* beschreibt u.a. den Startzeitpunkt des ersten Fensters

[4] T_1: Fenstergröße 5, $\lambda = 3$, $\beta = 2$; T_2: Fenstergröße 5, $\lambda = 1$, $\beta = 1$; die Fenstergröße wurde gewählt, um in T_1 teilerfremd zu λ und β zu sein, womit z.B. das Zusammenfallen der *Window-Close* und *Periodic*-Ereignisse verhindert wird.

den. Eine detaillierte Erklärung würde den Rahmen dieses Beitrags sprengen; festzuhalten ist aber, dass für alle ununterscheidbaren Klassen nachvollzogen werden konnte, warum sie nicht unterscheidbar sind. Dazu waren insgesamt etwa 10 verschiedene Tests mit unterschiedlichen Testdaten nötig. Die Testdaten zur Unterscheidung der Klassen sind dabei konfigurationsspezifisch. Um den Aufwand der Testdatenerstellung klein zu halten, sollte die Anzahl der Konfigurationen also klein gehalten werden.

4 Zusammenfassung und Ausblick

Föderierte Datenstromsysteme sind auf eine formale Beschreibung der angebundenen DSS angewiesen, um definierte Anfrageergebnisse erzielen zu können. Falls ein geeignetes semantisches Modell vorliegt, können diese durch *black-box*-Tests weitgehend automatisch ermittelt werden. Dies wurde am Beispiel eines komplexen Modells evaluiert.

Die Erweiterung um zusätzliche DSS und Modelle wird aktuell untersucht. Nach ersten Erkenntnissen ist dabei die Verwendung anderer DSS mit relativ geringem Aufwand möglich, während zur Verwendung eines anderen Modells zumindest Simulator und Testdaten weitgehend neuzuentwickeln sind.

Als Fernziel sollen die Portierung von Datenstromanwendungen und die Entwicklung föderierter DSS unterstützt werden. Daneben wird ein Nutzen für die Neu- oder Weiterentwicklung semantischer Modelle erwartet.

Literatur

[BCD+09] I. Botan, Y. Cho, R. Derakhshan, N. Dindar, L. Haas, K. Kim, C. Lee, G. Mundada, M. Shan, N. Tatbul, Y. Yan, B. Yun, and J. Zhang. Design and Implementation of the MaxStream Federated Stream Processing Architecture. Technical Report TR-632, ETH Zurich Department of Computer Science, June 2009.

[BDD+10] I. Botan, R. Derakhshan, N. Dindar, L. Haas, R. J. Miller, and N. Tatbul. SECRET: A Model for Analysis of the Execution Semantics of Stream Processing Systems. In *VLDB 2010*, pages 232–243, Singapore, September 2010.

[DLB+11] M. Daum, F. Lauterwald, P. Baumgärtel, N. Pollner, and K. Meyer-Wegener. Black-box Determination of Cost Models' Parameters for Federated Stream-Processing Systems. In *IDEAS 2011*, pages 226–232, Lisbon, Portugal, September 2011.

[DLF+10] M. Daum, F. Lauterwald, M. Fischer, M. Kiefer, and K. Meyer-Wegener. Integration of Heterogeneous Sensor Nodes by Data Stream Management. In Takahiro Hara, Vladimir I. Zadorozhny, and Erik Buchmann, editors, *Wireless Sensor Network Technologies for the Information Explosion Era*, volume 278 of *Studies in Computational Intelligence*, pages 139–172. Springer, 2010.